

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-324353

(43)Date of publication of application : 07.12.1993

(51)Int.Cl.

G06F 9/46

(21)Application number : 04-151449

(71)Applicant : NIPPON TELEGR & TELEPH CORP
<NTT>

(22)Date of filing : 19.05.1992

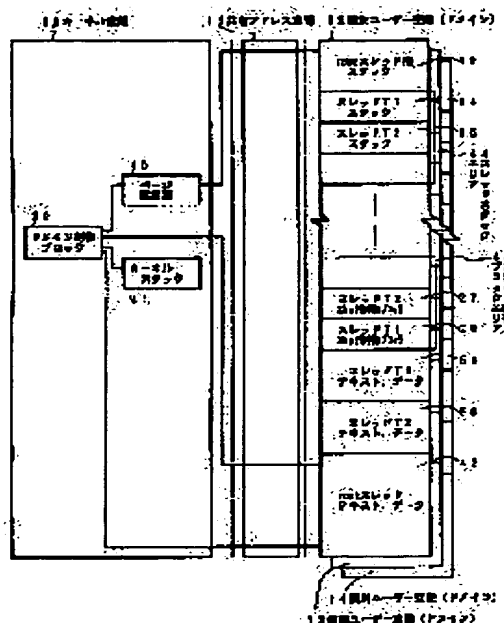
(72)Inventor : TANAKA SATOSHI
KUBOTA MINORU
MARUYAMA KATSUMI

(54) SYSTEM FOR CONTROLLING LIGHT-WEIGHT PROCESS

(57)Abstract:

PURPOSE: To evade the overhead of process control by means of change-over to a privilege mode by arranging a process control block in space which can be accessed by means of the non-privilege mode of logical address space.

CONSTITUTION: Individual user space 12 is address space which can be accessed by a user mode and has a high-order 2G byte address, where an application program is operated. Shared address space 11 is the only one space in a computer system having address which continues from whole individual user space 12 and kernel space 10, which can be accessed at the time of the user mode as it is. A process control structure body where information for managing and controlling the execution of light-weight process is held is arranged in logical address space which can be accessed by the non-privilege mode of logical address space, a stack being required for execution is adopted as the one only for the non-privilege mode and plural kinds of light-weight process are executed inside the logical address in parallel.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

BEST AVAILABLE COPY

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

<Partial Translation of Japanese Unexamined Patent
Publication No. H05-324353>

[0018] Fig. 2 shows a constitutional example of a logical address space.

[0019] A reference number 10 represents the kernel space having the size of one gigabyte. One can access the kernel space only when the processor is in a state of the kernel mode.

[0020] The reference numbers 11 and 12 represent a shared address space and an individual user space, respectively.

[0021] One can access the individual user space 12 only in the user mode. The individual user space has an address of high two gigabytes, in which an application program operates.

[0022] Multiple individual user spaces 12 may be provided.

[0023] The shared address space is an only address on the calculator system. It has a contiguous address from the respective individual user spaces and the kernel space. One can access the shared address space in the user mode.

[0024] The shared address space is used for arranging a message content of the interprocess message communications.

[0025] The individual user space 12 will be hereinafter referred to as a domain, and a lightweight process that operates on the domain will be referred to as a thread.

[0026] The domain and multiple threads that operates on the domain will be explained with reference to Fig. 3.

[0027] Domain control blocks 20 and 21 manages the domains 12 and 13.

[0028] In the domain control block, there are stored an address of a kernel stack in the domain, a page conversion table for constituting the logical address space, and a message reception matrix for message communications among threads, which will be explained below. A priority order for scheduling the domains, information such as runtime and the like, and resource information allocated in the units of domains is also stored in the domain control block.

[0029] Thread control blocks 26 and 27 in the domain 12 manage threads 22 and 23, which operate in the domain 12.

[0030] Thread control blocks 28 and 29 in the domain 13 manage threads 24 and 25, which operate in the domain 13.

[0031] The thread control blocks have context information of the thread, a text of the thread, data, information of the address of the stack, and a unique thread identifier (referred to as a thread ID).

[0032] Root threads 30 and 31 having a thread control function operates for controlling the threads in the respective domains.

[0033] The root threads 30 and 31 are generated by the kernel when the domains are generated. They initialize the domains.

[0034] The root threads performs the thread control function in the domain, such as the generation and deletion of the thread, the switch of the threads in the domain, etc.

[0035] The constitution of the domain and thread will be explained with reference to Fig. 1.

[0036] The domain control block 20 has a dedicated kernel stack 41 for the domain, the page conversion table 40, and the address information to the root thread 42.

[0037] In the domain 12 managed by the page conversion table 40, a stack area 45 for the root thread is provided

in the upper address. The text/data 42 for the root thread, the text/data 51 for the thread T1, and the text/data 50 for the thread T2 are provided in the lower-order address.

[0038] When generating the domain, a thread control block area 43 for the purpose of the thread control block is provided for pages for a certain number of thread control blocks is provided.

[0039] Furthermore, the stack area 45 for the root thread is followed by a certain size of the stack area 44 for the thread is provided.

[0040] The root thread having the domain process control function manages these areas.

[0041] The vacancy state of these areas, which are provided when the domain is generated, is managed, and if there are sufficient free area, they are reused.

[0042] On the contrary, if the size provided when the domain is generated exceeds, a memory space is provided by the kernel, and the page conversion table 40 is rewritten. Thus, these areas are expanded.

[0043] Thus, the dedicated kernel stack 41 for the domain and the page conversion table 40 are allocated to the domain 12, and the thread control block area 43, and then the stack area 44 for the thread are provided. Then, the control is given to the root thread 42.

[0044] The root thread 42 initializes the thread control block area 43 and the stack area 44 for the thread, and then the control is given to the application program. Then, the thread control blocks 26 and 27 for the threads T1 22 and T2 23 in the domain as well as the thread stacks 54 and 55 are allocated. Thus, the thread is generated.

[0045] Fig. 5 shows a processing flow for generating the domain.

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平5-324353

(43)公開日 平成5年(1993)12月7日

(51)Int.Cl.⁵

G 0 6 F 9/46

識別記号

3 4 0 B 8120-5B

庁内整理番号

F I

技術表示箇所

審査請求 未請求 請求項の数1(全12頁)

(21)出願番号 特願平4-151449

(22)出願日 平成4年(1992)5月19日

特許法第30条第1項適用申請有り 1991年11月21日 社
団法人電子情報通信学会発行の「電子情報通信学会技術
研究報告V o l . 91 N o . 333」に発表

(71)出願人 000004226

日本電信電話株式会社

東京都千代田区内幸町一丁目1番6号

(72)発明者 田中 聡

東京都千代田区内幸町一丁目1番6号 日

本電信電話株式会社内

(72)発明者 久保田 稔

東京都千代田区内幸町一丁目1番6号 日

本電信電話株式会社内

(72)発明者 丸山 勝己

東京都千代田区内幸町一丁目1番6号 日

本電信電話株式会社内

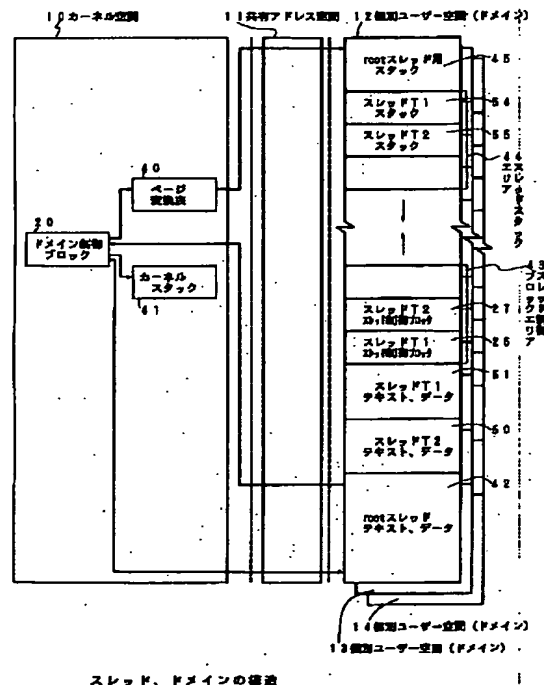
(74)代理人 弁理士 玉蟲 久五郎

(54)【発明の名称】 軽量プロセス制御方式

(57)【要約】

【目的】 必要メモリ量を小さくするとともにプロセス
制御のオーバーヘッドを避けることのできる軽量プロセ
ス制御方式を提供する。

【構成】 実行モードとして、少なくとも特権モード
(カーネルモード)と非特権モード(ユーザーモード)
の状態を持ち、複数の論理アドレス空間を制御する機能
を持つプロセッサから構成される計算機システムのプロ
セスを軽量化した軽量プロセス制御方式において、軽量
プロセスの実行を管理、制御するための情報を保持する
プロセス制御構造体を論理アドレス空間の非特権モード
でアクセス可能な論理アドレス空間に配置し、実行に必
要なスタックを非特権モード用だけとし、論理アドレス
内で複数の軽量プロセスが並行実行する。同時に多数の
プロセスがプロセッサ上で動作し、短時間に、多数のプ
ロセスが生成、消滅を繰り返し、複数のプロセスが互い
に情報をプロセス間通信しながら並行多重処理を進めて
いくアプリケーションプログラムに適用する、オーバー
ヘッドの小さいプロセス制御方式を構成する。



【特許請求の範囲】

【請求項1】 実行モードとして、少なくとも特権モードと非特権モードの状態を持ち、複数の論理アドレス空間を制御する機能を持つプロセッサから構成される計算機システムのプロセスを軽量化した軽量プロセス制御方式において、

前記軽量プロセスの実行を管理、制御するための情報を保持するプロセス制御構造体を論理アドレス空間の非特権モードでアクセス可能な論理アドレス空間に配置し、実行に必要なスタックを非特権モード用だけとし、論理アドレス内で複数の軽量プロセスが並行実行することを特徴とする、軽量プロセス制御方式。

【発明の詳細な説明】**【0001】**

【産業上の利用分野】本発明は、実行モードとして、少なくとも特権モードと非特権モードの状態を持ち、複数の論理アドレス空間を制御する機能を持つプロセッサから構成される計算機システムにおけるプロセスを軽量化した軽量プロセス制御方式に関し、特に同時に多数のプロセスがプロセッサ上で動作し、短時間に、多数のプロセスが生成、消滅を繰り返し、複数のプロセスが互いに情報をプロセス間通信しながら並行多重処理を進めていくアプリケーションプログラムに適用する、オーバーヘッドの小さいプロセス制御方式に関するものである。

【0002】

【従来の技術】従来使用されているオペレーティングシステムとして、UNIXの場合について、図7を用いて説明する。

【0003】図7は、UNIXにおけるプロセスの構造を示している。

【0004】UNIXの1プロセスは一つの論理アドレス空間に対応しており、論理アドレス空間88、89、90において、それぞれ一つのプロセスが動作する。

【0005】一つのプロセスに対応し、カーネル空間82内のプロセス構造体80がアロケートされ、スケジューリング情報、識別子、メモリなどの資源がUNIXカーネルによって管理される。

【0006】そして、ページ変換表81によってプロセスの動作する論理アドレス空間88が管理される。

【0007】論理アドレス空間88内の上位アドレスに、プロセスのシステムコール実行用のカーネルスタック87、プロセスのコンテキストを管理するためのu構造体86、アドレス空間内の下位アドレスに、プロセスのテキスト83とデータ84が配置される。

【0008】このように、UNIXのプロセスは、それを構成するためのリソースが多く、それらを確保したり、あるいは解放するためのオーバーヘッドは非常に大きい。

【0009】そのため、UNIXのプロセスの生成と消滅、切り替えのオーバーヘッドが大きい。

【0010】特に、同時に多数の呼を実時間に処理する必要のある交換処理をプロセスで構成した場合、プロセスは比較的小規模だが、その数が多い、1、2Kステップ走行することにメッセージ通信の事象待ちを生じ、コンテキストスイッチが頻繁に発生する、という特徴がある。

【0011】

【発明が解決しようとする課題】上記のような交換処理に従来のUNIXプロセスを適用した場合、プロセスの生成、消滅、コンテキストスイッチのオーバーヘッドのため、高い多重処理能力と実時間処理能力を同時に満足するようなシステムの構築は困難であり、必要なメモリ量も増大するという問題がある。

【0012】本発明の目的は従来の問題点を解決し、必要メモリ量を小さくするとともにプロセス制御のオーバーヘッドを避けることのできる軽量プロセス制御方式を提供することにある。

【0013】

【課題を解決するための手段】本発明は上記目的を達成するため、実行モードとして、少なくとも特権モードと非特権モードの状態を持ち、複数の論理アドレス空間を制御する機能を持つプロセッサから構成される計算機システムのプロセスを軽量化した軽量システム制御方式において、前記軽量プロセスの実行を管理、制御するための情報を保持するプロセス制御構造体を論理アドレス空間の非特権モードでアクセス可能な論理アドレス空間に配置し、実行に必要なスタックを非特権モード用だけとし、論理アドレス内で複数の軽量プロセスが並行実行することを特徴とする。

【0014】

【作用】本発明は、同時に多数のプロセスがプロセッサ上で動作し、短時間に、多数のプロセスが生成、消滅を繰り返し、複数のプロセスが互いに情報をプロセス間通信しながら並行多重処理を進めていくアプリケーションプログラムに適用する、オーバーヘッドの小さいプロセス制御方式で、プロセスの実行管理、制御するための情報を保持するプロセス制御構造体を論理アドレス空間の非特権モードでアクセスできる空間に配置し、プロセスは非特権モード用のスタック一本のみを持つ、論理アドレス空間内で並行実行可能な複数の軽量プロセスを構成する。

【0015】

【実施例】以下、本発明の実施例を図1を用いて説明する。

【0016】本実施例では、プロセッサは32bit幅のアドレスを取り扱い、プロセッサの実行モードとして、特権モードであるカーネルモードと、非特権モードであるユーザーモードを持ち、ページ変換表のアドレスをプロセッサの制御レジスタに設定しておくことにより、論理アドレスから物理アドレスへの変換を自動的に行なう

機能を持っているものとする。

【0017】アプリケーションとして、交換処理を考える。

【0018】図2に論理アドレス空間の構成例を示す。

【0019】10はカーネル空間で1Gバイトの大きさを持ち、プロセッサがカーネルモードの状態にある時のみアクセス可能な空間である。

【0020】11は共有アドレス空間、12は個別ユーザー空間である。

【0021】個別ユーザー空間12は、ユーザーモードでアクセス可能なアドレス空間で、上位2Gバイトのアドレスを持ち、アプリケーションプログラムが動作する。

【0022】個別ユーザー空間12は複数存在することができる。

【0023】共有アドレス空間11は計算機システム上唯一の空間で、すべての個別ユーザー空間、カーネル空間から連続したアドレスを持ち、ユーザーモードのままアクセス可能である。

【0024】共有アドレス空間11は、プロセス間メッセージ通信のメッセージ内容を配置するために使用する。

【0025】以下、個別ユーザー空間12をドメイン、ドメインで動作する複数の軽量プロセスをスレッドとよぶ。

【0026】図3を用いて、ドメインとその上で動作する複数のスレッドについて説明する。

【0027】ドメイン12、13は、カーネル空間10内のドメイン制御ブロック20、21で管理される。

【0028】ドメイン制御ブロックは、ドメインで持つカーネルスタックのアドレス、論理アドレス空間を構成するためのページ変換表、後述するスレッド間メッセージ通信のメッセージ受信行列、ドメインのスケジューリングのための優先順位、実行時間などの情報、ドメイン単位でアロケートされる資源情報などが格納される。

【0029】ドメイン12の中で動作するスレッド22、23は、ドメイン12内のスレッド制御ブロック26、27より管理される。

【0030】ドメイン13の中で動作するスレッド24、25は、ドメイン13内のスレッド制御ブロック28、29より管理される。

【0031】スレッド制御ブロックはスレッドのコンテキスト情報、スレッドのテキスト、データ、スタックのアドレスの情報、ユニークに決まるスレッド識別子(スレッドIDと呼ぶ。)を持つ。

【0032】各ドメイン内でのスレッド制御を行うため、スレッド制御機能を持つrootスレッド30、31が動作する。

【0033】このrootスレッド30、31は、ドメイン生成時にカーネルにより生成され、ドメインの初期設定

を行なう。

【0034】そして、スレッドの生成、削除、ドメイン内のスレッドの切り替えなどのドメイン内のスレッド制御機能はrootスレッドが行なう。

【0035】図1を用いて、ドメインとスレッドの構造について説明する。

【0036】ドメイン制御ブロック20は、ドメイン専用のカーネルスタック41、ページ変換表40、rootスレッド42へのアドレス情報を持つ。

【0037】ページ変換表40で管理されるドメイン12では、アドレスの上位にrootスレッド用のスタックエリア45が確保され、下位アドレスにrootスレッドのテキストとデータ42、スレッドT1用のテキストとデータ51、スレッドT2用のテキストとデータ50が置かれる。

【0038】ドメイン生成時には、スレッド制御ブロックのためのスレッド制御ブロックエリア43が、一定個数のスレッド制御ブロック分のページが確保される。

【0039】また、rootスレッド用のスタックエリア45の続きには、スレッド用のスタックエリア44が一定の大きさ分確保される。

【0040】これらの領域は、ドメイン内プロセス制御機能をもつrootスレッドにより管理される。

【0041】ドメイン生成時に確保されたこれらの領域は、空塞状態が管理され、十分な空き領域がある場合には再利用される。

【0042】しかし、ドメイン生成時に確保された大きさを越えた場合には、カーネルによりメモリ空間が確保され、ページ変換表40が書き換えられ、これらのエリアは拡張される。

【0043】ドメイン制御ブロック20、ドメイン専用のカーネルスタック41、ページ変換表40がドメイン12用にアロケートされ、スレッド制御ブロックエリア43、スレッド用のスタックエリア44が確保されたあと、rootスレッド42に制御が渡される。

【0044】rootスレッド42はスレッド制御ブロックエリア43、スレッド用のスタックエリア44の初期設定を行ない、アプリケーションプログラムに制御が渡り、ドメイン内のスレッドT1 22、T2 23のスレッド制御ブロック26、27、および、スレッドのスタック54、55がアロケートされ、スレッドが生成される。

【0045】図5に上記のドメイン生成時の処理フローを示す。

【0046】本実施例でのスレッド間メッセージ通信について説明する。

【0047】ここでは、交換処理に適用できるように、スループットが高く、遅延時間の短いメッセージ通信機構を実現するために、通信するデータを共有空間上におき、そこへのポインターをスレッド間で通信する。

【0048】共有空間上の通信データをメッセージバッファと呼ぶ。

【0049】メッセージを送信しようとするスレッドは、共有空間上のメッセージバッファを捕捉し、そこにメッセージの内容を書き込む。

【0050】受信スレッドのスレッドIDを指定して、そのメッセージを送信する。

【0051】送信後は、送信スレッドはメッセージバッファに対してアクセスすることができない。

【0052】スレッドはスレッド制御ブロックにメッセージの受信キューを持っていて、スレッドに到着したメッセージは、到着順に受信キューにキューイングされる。

【0053】受信スレッドは、自分の受信キューから到着しているメッセージバッファを一つ選択して受信する。

【0054】ここでは、通信網内の1交換システムをノードと呼び、同じノード内、あるいは異なるノードに存在する二つのスレッド間でメッセージを通信して交換処理を行なう場合を例にして説明する。

【0055】ノード間の通信はカーネル内にあるノード間通信制御機能が行なうものとする。

【0056】図4で、本実施例でのメッセージ通信処理について具体的に説明する。

【0057】メッセージ通信処理は受信スレッドの存在する位置に応じて、以下の三つの場合がある。

【0058】1.受信スレッドが同じドメイン内に存在する場合。

【0059】スレッド22からスレッド23へ、メッセージバッファ60を送信する場合を説明する。

【0060】スレッド22はメッセージバッファ60をアロケートし、通信内容を書き込む。

【0061】メッセージ送信では、スレッド23のスレッド制御ブロック27のメッセージ受信キューにメッセージバッファ60がキューイングされる。

【0062】2.受信スレッドが同一ノード内の別ドメインに存在する場合。

【0063】スレッド22からスレッド24へ、メッセージバッファ61を送信する場合を説明する。

【0064】スレッド22がメッセージバッファ61を送信する時、スレッド制御ブロック28に直接メッセージをキューイングできないので、スレッド24の存在するドメイン13のドメイン制御ブロック21にメッセージをキューイングする。

【0065】ドメイン13がプロセッサの使用権を獲得して、ドメイン13に制御が切り替わる時に、ドメインに到着しているメッセージがドメイン13内のスレッド24のスレッド制御ブロック28にキューイングされる。

【0066】3.受信スレッドが別ノードに存在する場

合。

【0067】スレッド22からスレッド4へメッセージバッファ62を送信する場合を説明する。

【0068】受信スレッドIDから、受信スレッドは別ノードに存在することがわかるので、自分のノードのノード間通信制御機能に、メッセージの通信を依頼して、リターンする。

【0069】送信側ノード間通信制御機能はメッセージバッファ62の内容を受信ノードに送信する。

【0070】受信側ノード間通信制御機能はメッセージバッファ63を確保し、送信側ノード間通信制御機能から送られてきたメッセージをそこに格納する。

【0071】そして、受信側ノード間通信制御機能は受信スレッド4の存在するドメイン7のドメイン制御ブロック6にメッセージをキューイングする。

【0072】ドメイン制御ブロック6にキューイングされたメッセージバッファ63は、ドメイン7にプロセッサの実行が移される時に、受信スレッド4のスレッド制御ブロック5にキューイングされる。

【0073】図6に、一例としてメッセージ送信の場合の処理フローを示す。

【0074】メッセージ受信キューは、スレッド制御ブロックあるいはドメイン制御ブロックを受信キューの先頭とし、メッセージバッファの双方向リンクを組むことによって、構成される。

【0075】特に、スレッド制御ブロックのメッセージ受信キューのアクセスには排他制御を行なう。

【0076】先のメッセージ通信処理で示したように、別ノード上のスレッドにメッセージを送信する場合、ノード間メッセージ通信機能にメッセージ送信を依頼した後で、スレッドには制御が戻される。

【0077】その後、受信スレッドが存在しないなどの通信エラーが発生した場合、そのエラーの発生を非同期に知らせ、優先的に処理する必要がある。

【0078】そのため、非同期に発生するエラーはメッセージとしてスレッドに通知される。

【0079】このメッセージを緊急メッセージと呼ぶ。

【0080】緊急メッセージは、通常のメッセージの受信キューの先頭にキューイングされ、優先的に処理される。

【0081】スレッドのメモリエラーなど、エラーが発生したスレッドが引続き実行することが不可能なエラーの場合、緊急メッセージはrootスレッドに送信される。

【0082】rootスレッドは、ドメイン生成時には、ドメイン内スレッド制御機能の初期設定をおこない、その後は、次のような処理を繰り返すスレッドとして、実現される。

【0083】1.rootスレッドに到着している緊急メッセージがあれば、それを処理する。

【0084】2.ドメインに到着しているメッセージを受

信スレッド制御ブロックにキューイングする。

【0085】3.次に制御を移すスレッドを選択する。

【0086】4.メッセージ受信待ちの状態に遷移し、選択されたスレッドに制御をうつす。

【0087】

【発明の効果】以上述べたように本発明は、プロセス制御ブロックを論理アドレス空間の非特権モードでアクセスできる空間に配置するため、プロセスの生成、消滅は、特権モードで実行する必要がないので、特権モードへの切り替えによる、プロセス制御のオーバーヘッドを避けることができる。

【0088】またプロセスの実行に必要なスタック、プロセス制御ブロックは、すべて非特権モードでアクセスできる空間にあるため、その確保、解放のオーバーヘッドを小さくすることができる。

【0089】さらに一つのプロセスを構成するためのカーネルのリソースが少なくできるので、同時に多数のプロセスを制御する必要のある交換システムに適用した場合、必要メモリ量を小さくすることができる。

【0090】また同じ論理アドレス空間に属するプロセス間でのコンテキストスイッチ、プロセス間メッセージ通信において、特権モードに遷移する必要がないので、これらのオーバーヘッドを小さくできる。

【図面の簡単な説明】

【図1】本実施例におけるドメインとスレッドの構造を示したものである。

【図2】実施例における論理アドレス空間の構成を示した図である。

【図3】実施例におけるドメインとスレッドの制御の関係を示した図である。

【図4】実施例におけるプロセス間メッセージ通信処理の概要を説明する図である。

【図5】図1に対するドメイン生成時の処理フローチャートである。

【図6】図4に対するメッセージ送信の場合の処理フローチャートである。

【図7】従来の技術を説明するための図である。

【符号の説明】

9, 10, 82 カーネル空間

8, 11 共有アドレス空間

7, 12, 13, 14, 15 個別ユーザー空間（ドメイン）

6, 20, 21 ドメイン制御ブロック

4, 22, 23, 24, 25 軽量プロセス（スレッド）

5, 26, 27, 28 29 スレッド制御ブロック

30, 31, 42 rootスレッド

60, 61, 62, 63 メッセージバッファ

40, 81 ページ変換表

41, 87 カーネルスタック

43 スレッド制御ブロックエリア

44 スレッドスタックエリア

45 rootスレッド用スタック

50 スレッドT2テキスト、データ

51 スレッドT1テキスト、データ

54 スレッドT1スタック

55 スレッドT2スタック

80 プロセス構造体

83 テキスト

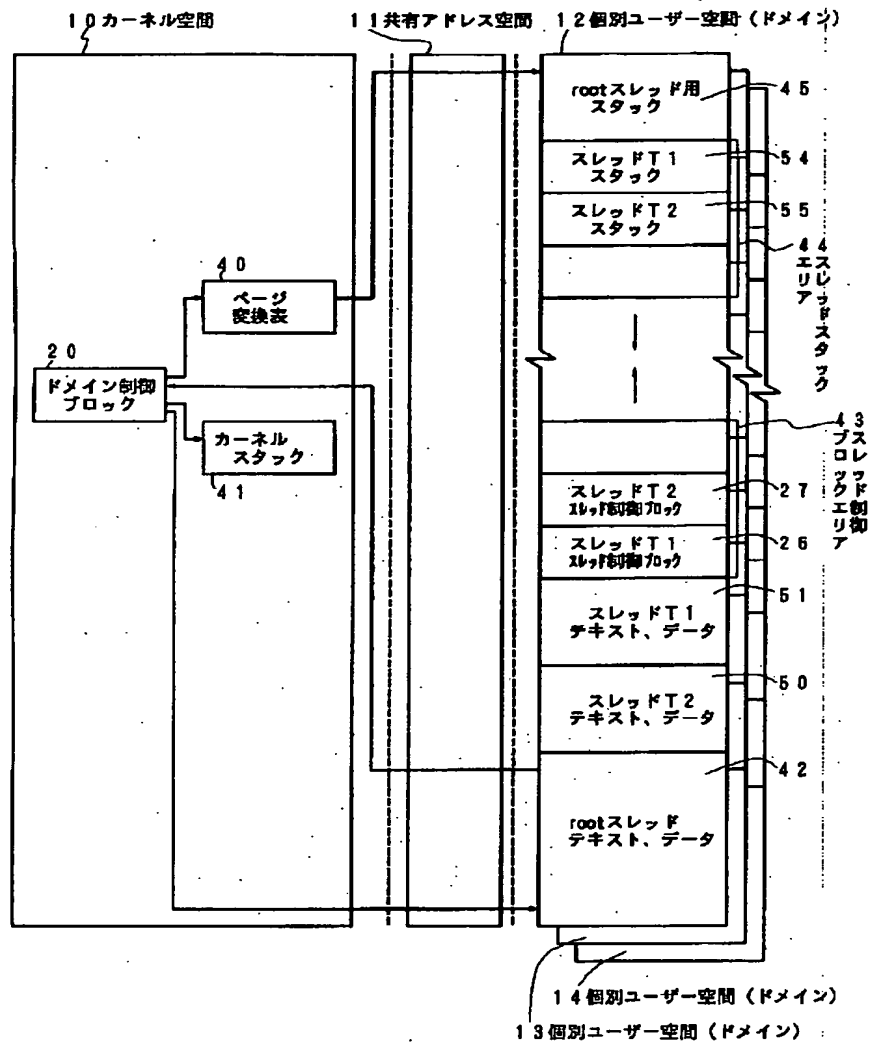
84 データ

85 ユーザースタック

86 u構造体

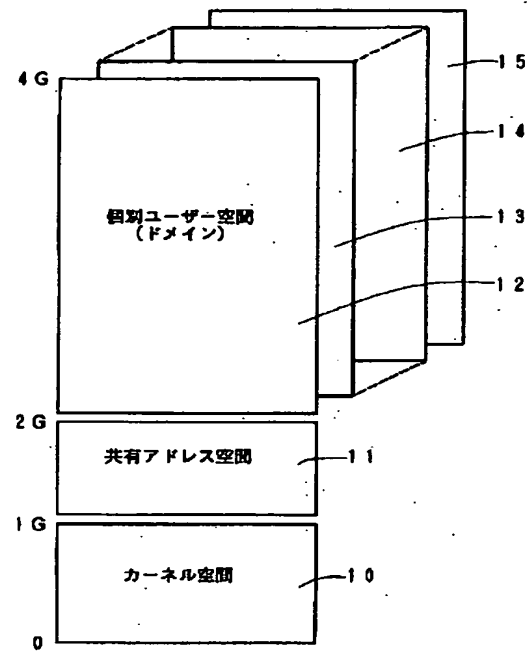
88, 89, 90 論理アドレス空間

【図1】



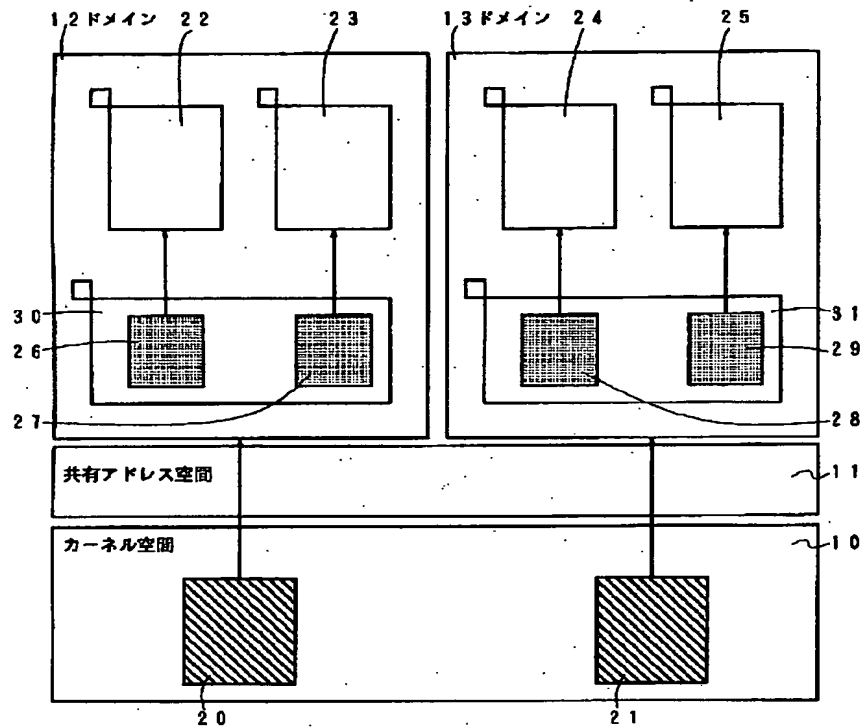
スレッド、ドメインの構造

【図2】



論理アドレス空間の構成

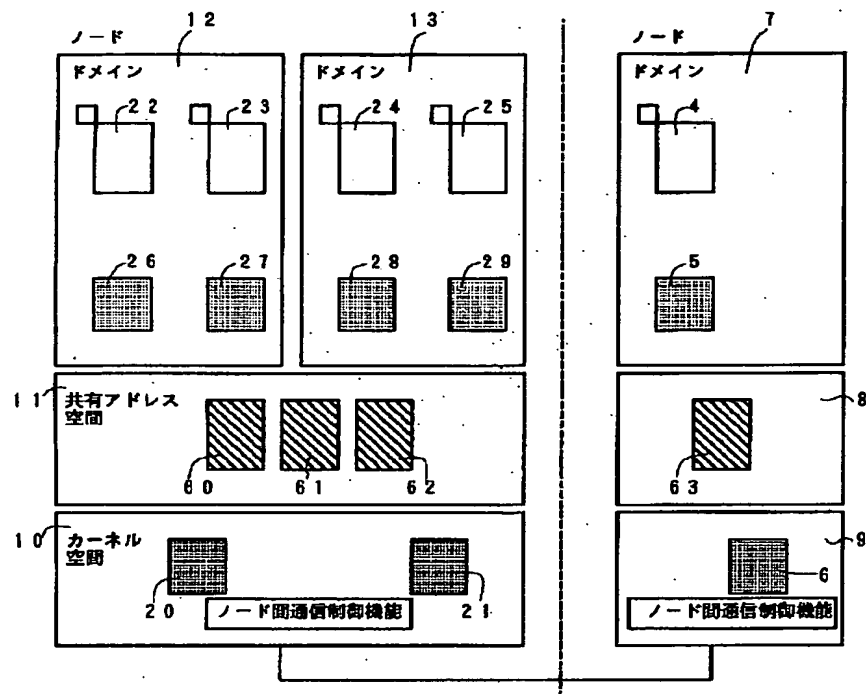
【図3】



20, 21...ドメイン制御ブロック
 22, 23, 24, 25...スレッド
 26, 27, 28, 29...スレッド制御ブロック
 30, 31...rootスレッド

スレッド、ドメインの制御

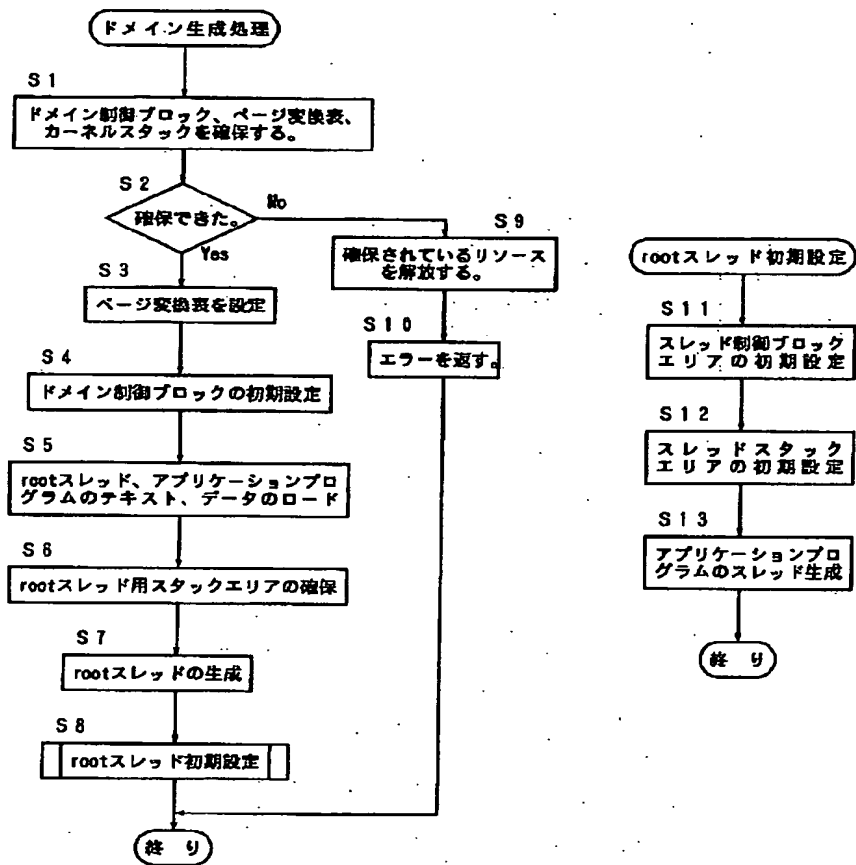
【図4】



8, 20, 21...ドメイン制御ブロック
 4, 22, 23, 24, 25...スレッド
 5, 26, 27, 28, 29...スレッド制御ブロック
 60, 61, 62, 63...メッセージバッファ
 8...共有アドレス空間
 9...カーネル空間

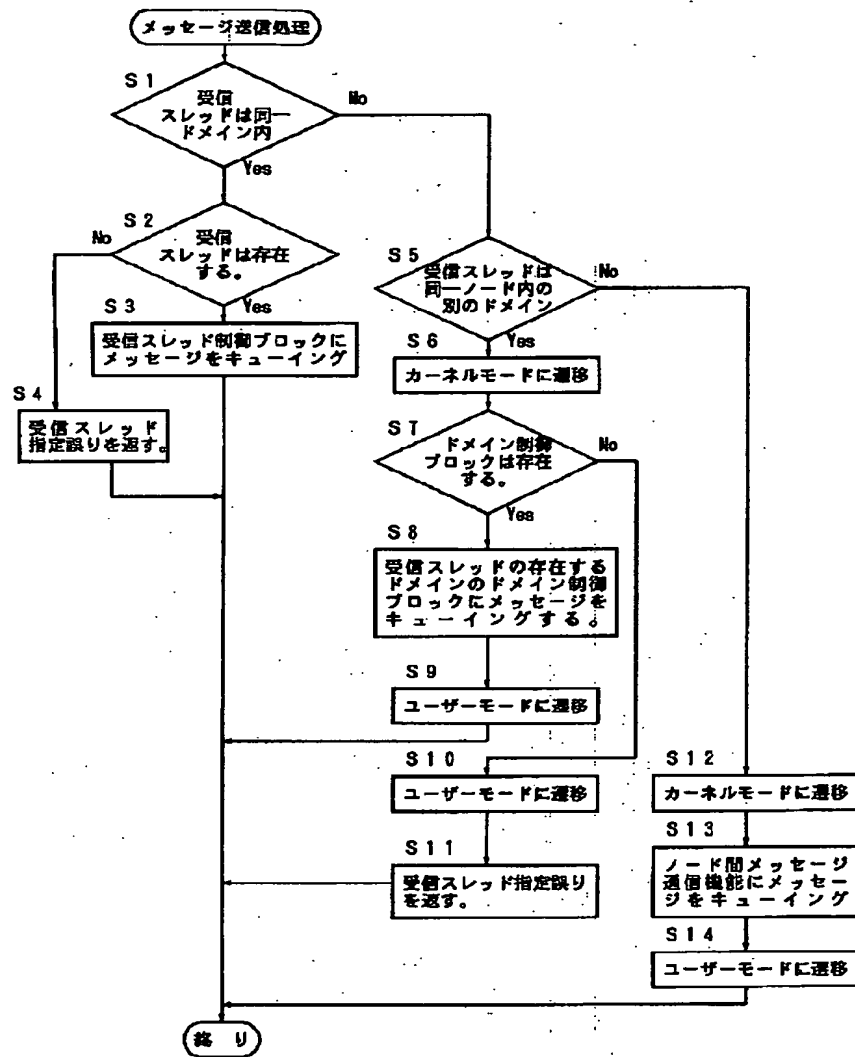
メッセージ通信機能

【図5】

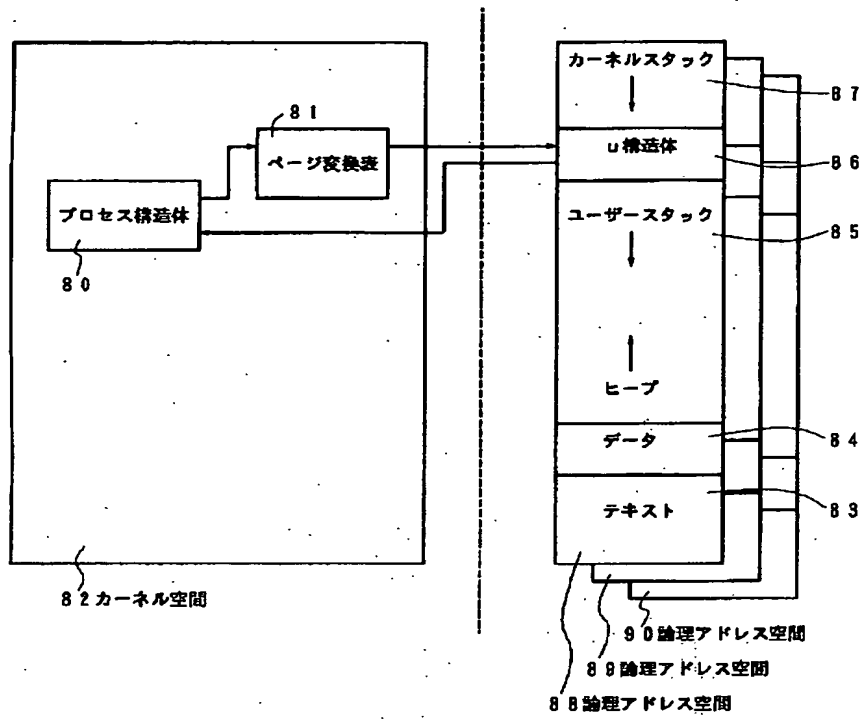


ドメイン生成時の処理フロー

【図6】



【図7】



UNIXプロセスの構造

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.